Multi-Agent System for Resource Reliability

PI: Tamitha Carpenter

Final Report

Reporting Period: April, 1999 through December, 1999

Contract Effective: April 28, 1999

December 20, 1999

Sponsored by:

Defense Advanced Research Projects Agency (DOD)
(controlling DARPA Office)

ARPA Order No. D611, Amdt 56

Issued by US Army Aviation & Missile Command Under
Contract Number: DAAH01-99-C-R137

Contractor:

Stottler Henke Associates, Inc. (SHAI)
1660 So. Amphlett Blvd., Suite 350
San Mateo, CA 94402
(650) 655-7242

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 20, 1999	3. REPORT TYPE AND DATES COVERED Final Report 4/28/99-12/29/99

4. TITLE AND SUBTITLE Multi-Agent System for Resource Reliability	5. FUNDING NUMBERS DAAH01-99-C-R137
6. AUTHORS Tamitha Carpenter	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stottler Henke Associates, Inc. 1660 South Amphlett Blvd., Suite 350 San Mateo, CA 94402	8. PERFORMING ORGANIZATION REPORT NUMBER Report No. 192: Reliability Final Report
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES NONE	
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited;	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Report developed under SBIR contract. In this Phase I SBIR research we demonstrated the feasibility of a unique system for maintaining network resource reliability through a decentralized multi-agent architecture by prototyping key features of the system. Our approach differs from other computer security tools in a number of ways. First, our system does not rely on the presence of a central analysis unit. Instead, individual agents monitor individual hosts and/or local portions of the network, communicating between agents when needed. This design is not only easily scalable, it enables the agents to perform local responses, even when much of the network is compromised. Second, unlike many intrusion detection programs which have hard-coded responses to known attacks, our system can respond to both known attacks and unexplained anomalies. Third, our agents are able to reconcile differing local view through data fusion, thus enabling the agents to respond to network degrading events with increasing accuracy as information is collected and aggregated from other agents. Phase I research and development of a prototype for network resource reliability has laid the groundwork for the Phase II implementation of MASRR, a Multi-Agent System for Resource Reliability, and its eventual commercialization.

14. SUBJECT TERMS SBIR Report, Information Warfare, Computer Security, Active Networks, Multi-Agent Systems	15. NUMBER OF PAGES 27
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

Multi-Agent System for Resource Reliability
Final Report

1 Introduction

The rapid growth of computer networks over the past ten years has resulted in a highly complicated operating environment susceptible to a variety of attacks and malfunctions capable of compromising system reliability at all levels. Whether network resource reliability is lost due to malicious behavior or simple device failure, the cost of this loss can be astronomical, both in terms of lost productivity and the possible theft of sensitive documents.

Maintaining network system reliability is a difficult problem, given that modern computer networks involve many different computer platforms and network elements interacting over a variety of network protocols and implementations of those protocols, all combinable in a virtually infinite number of configurations. These complicated environments lead to two major obstacles to maintaining network reliability: 1) ensuring that every host and every network element is properly configured at all times is a monumental task and is rarely, if ever, accomplished; and 2) any software solution for detecting and responding to network reliability issues must be sensitive to the variability that exists in these networks.

Another difficulty with maintaining network reliability stems from flaws in the designs and implementations of operating systems and networking protocols. TCP/IP (Transmission Control Protocol/Internet Protocol), the primary networking protocol on the internet and a prevalent protocol on smaller networks, is a prime example. In a defining paper for the computer security industry, [Bellovin, 1989] identifies a number of problems inherent with the *Transmission Control Protocol/Internet Protocol* (TCP/IP) protocol suite, which open networks to security breaches through attacks such as "address spoofing", the "smurf" attack [CERT CA-98.01], and others. Particular implementations of TCP/IP have also opened vulnerabilities to Denial of Service (DoS) attacks such as the "land" attack and the "Teardrop" attack [CERT CA-97.28].

New vulnerabilities that enable attacks similar to those mentioned above are constantly being found and exploited. In order to maintain network reliability in the face of these constantly evolving attacks, not to mention compensating for misconfigurations of operating systems and network elements, a software solution for maintaining network reliability is needed that not only works across heterogeneous networks and internetworks, but is also scaleable to a variety of network sizes and is able to continue functioning even when portions of the network are unavailable or have become compromised. This final report describes our Phase I SBIR efforts in defining such a solution.

1.1 Phase I Objectives

The objective of the Phase I research was to develop a multi-agent system for maintaining network resource reliability. All of the tasks that were set out in the Phase I proposal were successfully accomplished. The new capabilities that we have developed support decentralized, collaborative evaluation of network resource reliability and provide the foundation for developing techniques to respond to potential breaches in reliability and security. Phase I research and development have laid the groundwork for the Phase II implementation of a complete system for evaluating and maintaining network resource reliability, and its eventual commercialization. The primary goals of the Phase I research were to:

- **Develop the concept of a multi-agent system for resource reliability to a point where its effectiveness can be demonstrated through a proof-of-concept prototype.** In the Phase I proposal, this objective focused on the ability to maintain resource reliability in the face of Denial of Service (DoS) attacks. Our research in Phase I exposed several key facts about the nature of DoS attacks that forced us to change our concept of what our multi-agent system should do. Primarily, the nature of most network-based DoS attacks involves executing a single directive that takes advantage of some weakness in a network communication protocol or an operating system. The most effective defense against such an attack is to establish appropriate filters on firewalls, gateways, and routers. Because these DoS attacks execute so rapidly, any attempt to establish these filters during the course of an attack will fail, simply because the offending command will already be on its way to its designated target by the time it is detected.

Although our multi-agent system for resource reliability will not improve a networks resistance to known DoS attacks, it does provide several far-reaching benefits. When an unknown DoS exploit is used to attack the network, our system will be able to mitigate the side effects resulting from the attack. In addition, we have shown that our system will provide protection against other forms of attack, such as Worms (malicious programs that replicate themselves and spread to other machines in a network), and more subtle network problems, such as compromised routing tables. These issues are described more fully in Section 3 – "Phase I Investigation."

- **Obtain a comprehensive understanding of resource reliability in today's heterogeneous computer networks.** During Phase I, we performed a detailed investigation of DoS attacks, other methods for attacking networks and networked hosts, and non-attack incidents (e.g., misconfigured routing tables) that can negatively impact network resources. We also investigated the current state-of-the-art in network security software to determine how well network reliability is currently maintained. We found a variety of tools that detect the occurrence of known attacks on host computers, other tools that detect specific patterns in network traffic, and a few tools that provide a limited level of response to detected attacks. However, none of the current tools provide the breadth of functionality for detecting and responding to both known and anomalous network degrading events that is necessary for maintaining acceptable levels of resource reliability.
- **Develop the concepts necessary for effective information agents, decision agents, and enforcer agents.** During the course of the Phase I investigation, the architecture for this system changed dramatically. The Phase I proposal presented a system architecture with separate agents for collecting information, forming decisions, and enforcing security policies. In the prototype developed for this Phase I project, the functions for these three agents were merged into a single agent or "Reliability monitor". Each agent gathers local information (e.g., system load, memory usage, etc.) and possibly data from a network sniffer, and uses this information to choose an "action template." The action template includes directives for communicating with other reliability agents and for responding to anomalous behavior.
- **Establish a means for the multi-agent system to work together and result in reliable resource availability across computer networks.** SHAI investigated a variety of approaches for communication between the individual agents. The communication system developed focuses on three issues: 1) aggregating information from multiple agents without a need for a centralized control unit; 2) performing effective communication without negatively impacting network performance; and 3) ensuring that reliability issues are fully addressed, even when the agent that originated the request breaks down (i.e., the computer on

which it operates goes down or loses network connectivity). Issues in secure communication between agents were considered, but implementation was deferred to Phase II.

2 Phase I Investigation

The objective of the Phase I prototype was to investigate methods for maintaining network resources in the face of network degrading occurrences, such as Denial of Service (DoS) attacks, other types of attacks, misconfigured routing tables, etc. An important focus of this work has been to insure the scalability of the resulting system, so that it would as easily increase the reliability of a simple 30 node local area network (LAN) as a large 10,000 node Wide Area Network (WAN). In addition to assuring scalability, the resulting system must also be robust in the face of a degraded network.

The starting point for our Phase I investigation was to examine the possibilities for employing *local agents* to monitor the network and respond appropriately when a network degrading event occurs. Our proposed design was inspired in part by [Tomlin, Pappas, and Sastry, 1997], which describes an elegant system for conflict resolution for air traffic management. Their solution was based on agents employing *local rules* of operation for the global effect of resolving conflicts in flight plans.

Designing a system of multiple agents distributed across a network provides much of the needed scalability, a properly designed agent need only process data that is local to its host computer or other network element (e.g., router, firewall, etc.). In order to build a multi-agent system that is also robust in the face of a degraded network, we also chose to design a completely *decentralized* system. That is, our Multi-Agent System for Resource Reliability (MASRR) does not rely on any notion of a centralized control program, since communication with such a key element could not be guaranteed. Instead, we have designed MASRR to process local information, and communicate with other instances of MASRR as the need arises.

As we will describe in section 2.1, several safeguards ensuring that MASRR responds to important events are built into our design. Primarily, these safeguards allow for continued processing and response, even when certain network elements and their associated MASRR agent go down:

- When the initiating MASRR agent (i.e., the agent that first "notices" that a network degrading event may be occurring) determines that it needs information from its network peers, it will continue processing, even if some of its peers do not respond. Further, a lack of response from a peer may provide further evidence that a network degrading condition exists, and the initiating MASRR agent will respond appropriately.
- If the computer on which the initiating MASRR agent exists goes down, one of its network peers will assume responsibility for processing and responding to the situation. This provides a maximum level of robustness in the unpredictable domain of network management.

Each MASRR agent will have a "local view" of the network, encompassing concepts such as what is "normal" behavior, how current behavior differs from normal, which of its network peers may be effected under certain circumstances, and how to respond when the situation warrants a response. An important design consideration for MASRR agents is that they must be able to act in the face of incomplete information.

One final consideration of the design for MASRR is that it must operate on today's complex, heterogeneous networks. Figure 1 illustrates a sample operating environment for the MASRR agents. MASRR agents will be able to reside on a number of different platforms and operating systems, but communication between MASRR agents will be platform independent, ensuring maximum cooperation between agents. Instances of the MASRR agents (indicated by white circles) will be available on a number of network elements, but we cannot assume that all elements will be covered, particularly elements that are under external jurisdiction (e.g., an internet service provider).

The addition of MASRR monitoring and response will require very little additional hardware, since, in most cases, MASRR will not be computationally expensive and therefore MASRR agents will be able to reside on existing network elements. However, there are two special cases of MASRR agents. First, occasionally a dedicated MASRR host may be required if an extremely large amount of local data is being processed, such as is the case when a network sniffer is being used to monitor network traffic. Second, certain network elements will be monitored "remotely"; that is, from a computer that is separate from the network element. This allows non-host types of network elements (e.g., routers, hubs, etc.) to be under MASRR protection.

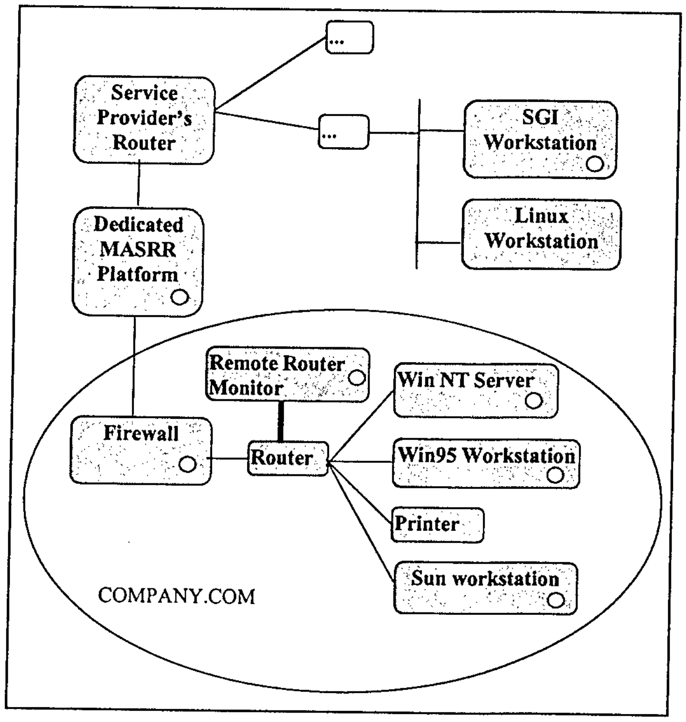


Figure 1. Operating environment description from the Phase I proposal, modified to reflect the system changes developed during the Phase I investigation.

2.1 Prototype Design

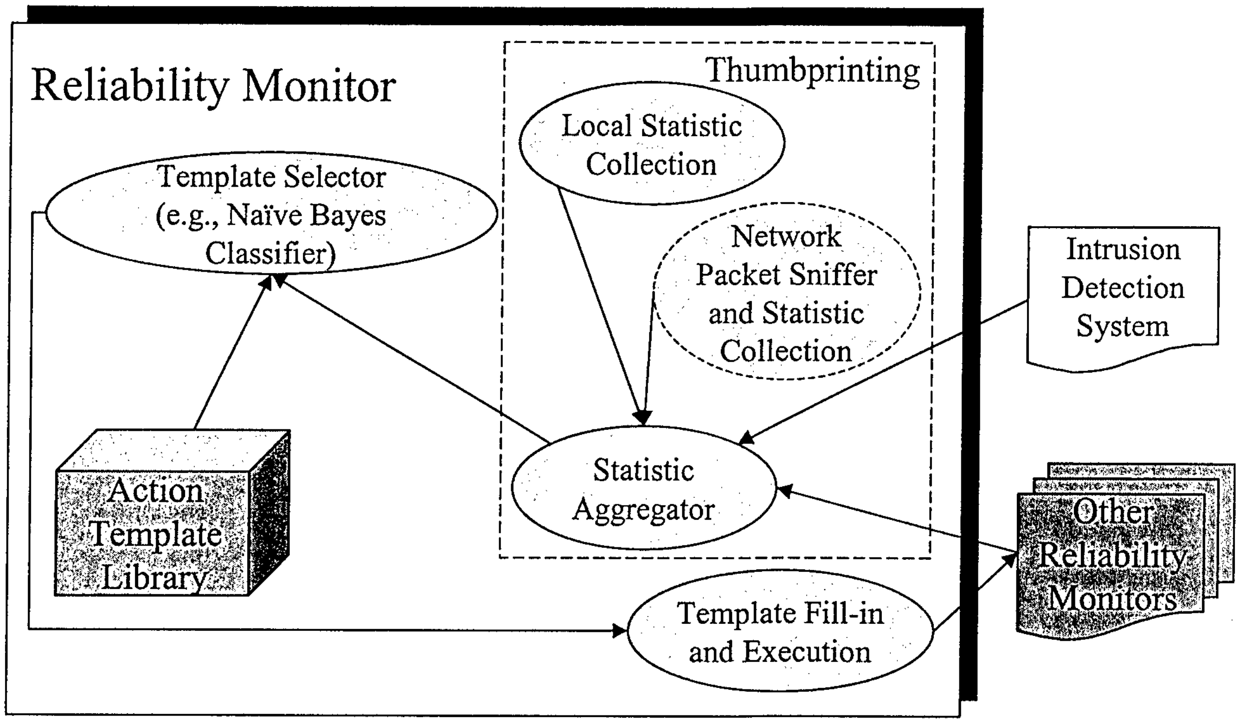

Reliability Monitor
Thumbprinting
Local Statistic Collection
Template Selector (e.g., Naïve Bayes Classifier)
Network Packet Sniffer and Statistic Collection
Intrusion Detection System
Action Template Library
Statistic Aggregator
Other Reliability Monitors
Template Fill-in and Execution

Figure 2. MASRR System Architecture.

Figure 2 gives an overview of the information flow within a single instance of a MASRR agent. For the Phase I prototype, we concentrated on four areas:

1. Developing a notion of "thumbprinting", including a comprehensive view of what constitutes "local statistics" and what kind of information can be gleaned from these statistics and from network traffic analysis.
2. Developing a method for controlling MASRR behavior and communication between agents through the use of an **Action Template Library**.
3. Elaborating a method for selecting templates from the Action Template Library.
4. Developing a robust method of communication between MASRR agents and between MASRR and SHAI's proprietary intrusion detection system, ICE.

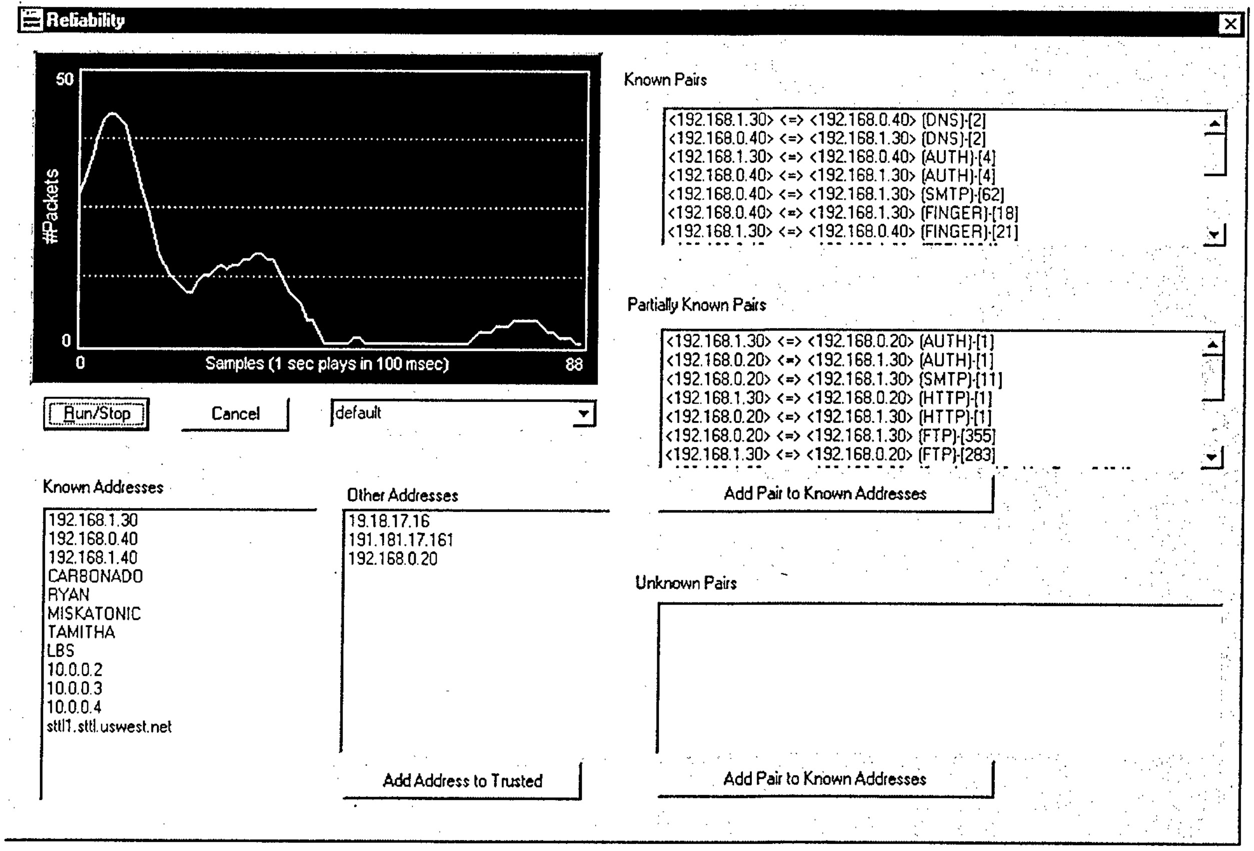
Thumbprinting – MASRR execution begins with continuous comparison of the current situation with local "thumbprints" – collections of data patterns that exemplify "normal" behavior under a variety of circumstances (e.g., Monday mornings, weekdays near midnight, etc.) The current situation is determined through the collection of local statistics (e.g., system load, memory usage, hard disk usage, etc.). Additionally, certain instances of MASRR also include statistic collection from a network packet sniffer (e.g., tcpdump). These specialized MASRR agents analyze network traffic to collect the following types of information:

- Number of accesses by trusted hosts, known hosts, and unknown hosts in a given time interval (e.g., every 20 seconds).

- Number of packets between known *pairs* of hosts, partially known pairs of hosts (i.e., one of the two addresses is known), and unknown pairs of hosts in a given time interval.
- Number of different types of packets (e.g., echo request packets, ftp packets, etc.) in a given time interval.

Figure 3 shows a screen shot of MASRR's analysis of a sample tcpdump file. The graph shows the overall network activity (i.e., the running count of all network packets). Other windows show a list of Known Addresses, Other Addresses encountered in the network activity, count of different types of network packets for 1) each Known Pair of network addresses, 2) each Partially Known Pair of network addresses, and 3) each Unknown Pair of network addresses.

In addition to local computer and network statistics, information regarding the probability of currently ongoing attacks may be provided by an intrusion detection system.


Reliability
50
0
#Packets
0
Samples (1 sec plays in 100 msec)
88
Run/Stop
Cancel
Add Pair to Known Addresses
Add Address to Trusted
Add Pair to Known Addresses

default

Known Pairs

```
<192.168.1.30> <=> <192.168.0.40> [DNS]-[2]
<192.168.0.40> <=> <192.168.1.30> [DNS]-[2]
<192.168.1.30> <=> <192.168.0.40> [AUTH]-[4]
<192.168.0.40> <=> <192.168.1.30> [AUTH]-[4]
<192.168.0.40> <=> <192.168.1.30> [SMTP]-[62]
<192.168.0.40> <=> <192.168.1.30> [FINGER]-[18]
<192.168.1.30> <=> <192.168.0.40> [FINGER]-[21]
```

Partially Known Pairs

```
<192.168.1.30> <=> <192.168.0.20> [AUTH]-[1]
<192.168.0.20> <=> <192.168.1.30> [AUTH]-[1]
<192.168.0.20> <=> <192.168.1.30> [SMTP]-[11]
<192.168.1.30> <=> <192.168.0.20> [HTTP]-[1]
<192.168.0.20> <=> <192.168.1.30> [HTTP]-[1]
<192.168.0.20> <=> <192.168.1.30> [FTP]-[355]
<192.168.1.30> <=> <192.168.0.20> [FTP]-[283]
```

Known Addresses

```
192.168.1.30
192.168.0.40
192.168.1.40
CARBONADO
RYAN
MISKATONIC
TAMITHA
LBS
10.0.0.2
10.0.0.3
10.0.0.4
sttl1.sttl.uswest.net
```

Other Addresses

```
19.18.17.16
191.181.17.161
192.168.0.20
```

Unknown Pairs

Figure 3. Analysis of network packets, showing a graph of overall network activity, plus counts of packet types sent between each unique pair of addresses.

Action Template Library – Deviations from stored thumbprints (in combination with statistics aggregated from other MASRR agents) are used to choose an **action template** from the action template library. Action templates determine how a MASRR agent should respond in particular circumstances. Responses include:

- Collect more statistics.

- Communicate with other MASRR agents.
- Communicate with a system administrator (e.g., "router 2" is not responding).
- Establish emergency security policies (e.g., establishing more restrictive filters on routers, disabling non-essential network services, increasing intrusion detection levels, etc.)

A conceptual drawing of action template use is shown in Figure 4. In this picture, a MASRR agent in LAN7 has selected an action template that specifies communication should be instigated with LAN4, LAN6, and LAN12, as well as Router3, Router8, and Bridge2. Only LAN4, LAN12, and Router3 have deviations from their own thumbprint information, which is returned to LAN7. The originating MASRR agent in LAN7 aggregates the returned information by "filling in" the action template. Depending on the actual content, the action template may specify that a new template should be selected.

Template Selector – A key aspect of the action template library is that action templates can be chosen <u>even with limited information</u>. This is important for two reasons. First, it is likely that "complete information" describing most network degrading events would be nearly impossible to collect, since some of the activity will occur outside of MASRR's jurisdiction. Second, even if complete information were available, collecting all of it would take time, but MASRR must be able to respond quickly when it detects such an event, in order to mitigate its effects before more of the network is affected.

To achieve template retrieval using partial information, we have designed a Template Selector, which uses classification techniques, such as a Naïve Bayes Classifier (e.g., [Keogh & Pazzani]), to find a partial match between available information and the action template indices. Our choice of the Naïve Bayes Classifier is based on the fact that this classification technique does a particularly good job of determining classifications with incomplete information. Template Selection occurs when the partial match between the classification and the template indices surpasses a potentially tunable threshold.

Additionally, the quality of this match will also influence how the agent uses the action template. The priority will be to ensure overall network stability by mitigating possible threats. However, in a situation where the match is less than ideal, information collection will continue, and, if the new information that is collected contradicts the previously selected action template, the previously performed response will be retracted. This policy could potentially cause unnecessary restrictions on the network, but only for very short periods of time. This inconvenience is far outweighed by the benefit of limiting damage when an actual network degrading event has occurred.

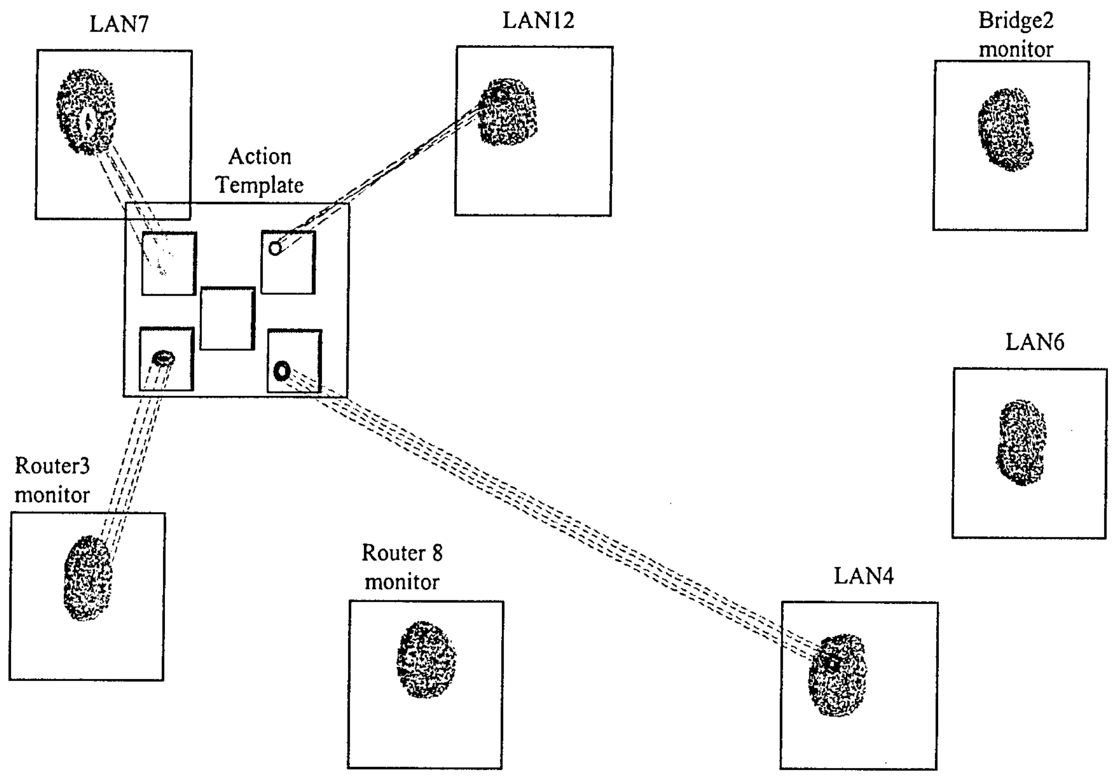


Figure 4. A conceptual picture of action template fill-in.

Communication Between Agents – The action templates specify what information should be shared with other MASRR agents, and which MASRR agents should receive that information. Additionally, these templates tell the other MASRR agents what information the originating agent is requesting. As more information is collected, the action templates that are selected have potentially wider communication patterns. For an individual host, a typical communication progression might be: 1) broadcast to other hosts in this LAN; 2) communicate with local routers; 3) local routers communicate with certain peer routers and other LANs.

Each MASRR agent has built-in safeguards, ensuring that network degrading events receive an appropriate response, even if key network elements become unavailable, while avoiding excessive overhead related to communication between agents and overly redundant processing. These safeguards include:

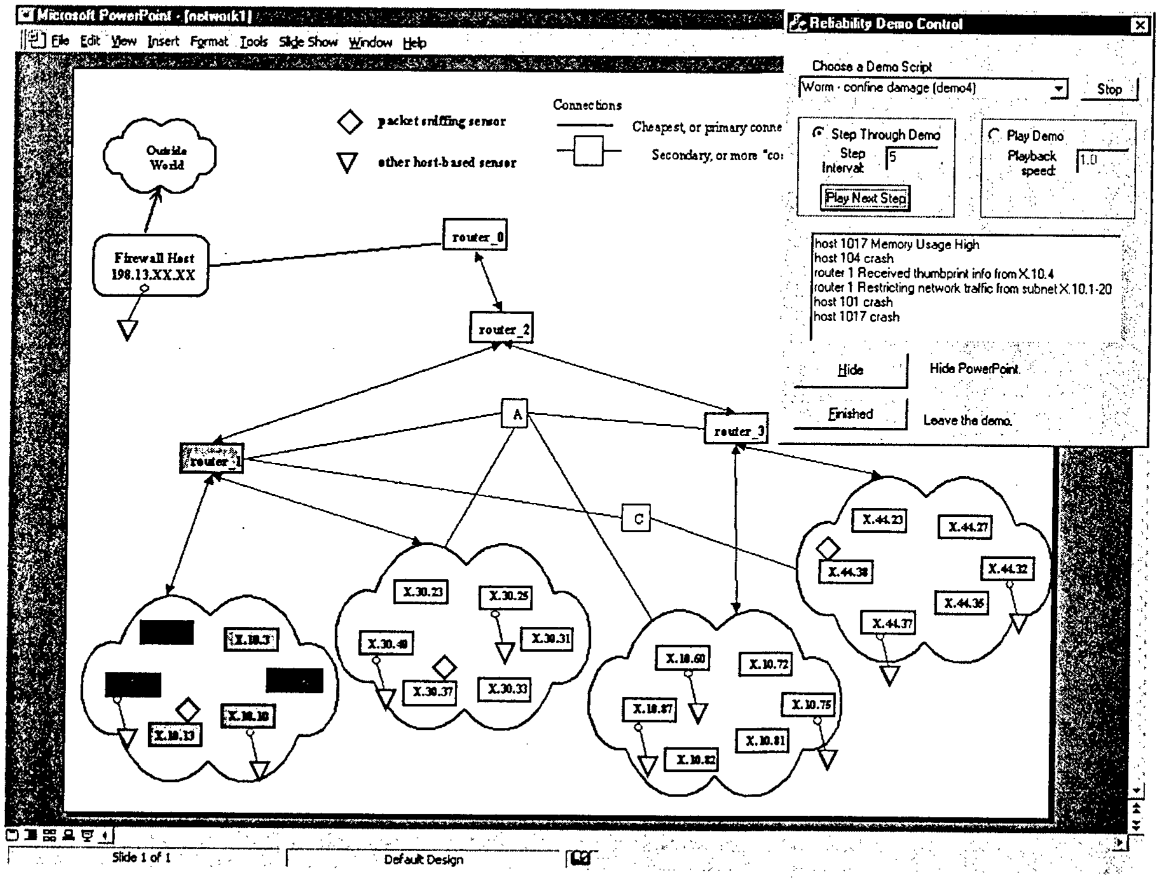
- The representation of the information exchanged between MASRR agents is very small, thus ensuring that MASRR will not contribute to further network degradation.
- When multiple MASRR agents request information almost simultaneously, the affected MASRR agents examine the timestamp of the requests. (For the prototype, MASRR simply assumes that the clocks for all MASRR are in sync.) The oldest request will receive responses, while the other requests will be retracted. Information from the retracted requests will be included in the reply to the oldest request.
- All MASRR messages are acknowledged, so that MASRR agents have up-to-date information about the status of their peers.

- If the originating MASRR agent goes down, a peer takes responsibility for continuing analysis and diagnosis.
- Lack of response and/or acknowledgement can act as negative evidence.

2.1.1 Demonstration Scenario – MASRR's response to a Worm-type attack

Figure 5 shows MASRR's behavior in response to a Worm-type attack (similar to the Internet Worm) occurring in a sub-network. This sequence begins when the MASRR agent for one of the machine's in this sub-network detects anomalous behavior. The MASRR agent broadcasts a message to the other hosts in this sub-network. Meanwhile, one of the other machines in the sub-network is infected by the worm, and ICE (the intrusion detection system) detects an attempt to access its host machine through a known "sendmail" bug. These machine's respond to the originating machine with the relevant information, and the other machine's respond with "no information."

When the originating MASRR agent receives the responses, it aggregates the information and sends a message to "router 1". Meanwhile, ICE detects an attempt to access its host machine through another known vulnerability, this time in the "finger" program. Another machine in the sub-network is infected, and the original machine crashes because the Worm has consumed too many resources.



```
host 1017 Memory Usage High
host 104 crash
router 1 Received thumbprint info from X.10.4
router 1 Restricting network traffic from subnet X.10.1-20
host 101 crash
host 1017 crash
```

Figure 5. Illustration of Reliability's approach to limiting the damage from a Worm-type attack.

When the MASRR agent on "router 1" receives the message regarding the anomalous behavior in the sub-network, it responds by restricting network access to the sub-network, thus preventing the worm from spreading further. It then sends messages to the other routers in the network so that appropriate preemptive measures can be taken. A message is also sent to the system administrator, so that repairs to the sub-network can be scheduled immediately.

2.2 Description of Methodologies

We approach the development of the MASRR system with considerable experience with computer network security, a number of innovative ideas, and extensive expertise in the field of AI and machine learning. We will not inappropriately impose a solution or methodology on the problem. Instead, our goal is to first thoroughly understand the complexities of the particular domain, and then, using appropriate techniques, tailor a solution to the problem. In this section, we will expand upon the innovative solutions mentioned earlier.

2.2.1 Multi-Agent Systems

There are many different types of multi-agent systems. One popular view is that a multi-agent system is a society of software agents which operate on the behalf of some operator (human or another software agent) to achieve some goal. Though perfectly valid, this type of multi-agent system is not relevant to this project. We are concerned with building agents whose coordinated, autonomous activity, will prove useful in decentralized, reactive, control applications (i.e., MASRR). As such, our approach is more focused on the multi-agent paradigm as a software engineering methodology for designing and implementing such systems. [Bradshaw, 1997] gives a good overview of a variety of types of software agents and multi-agent systems.

A multi-agent system is a federation of agents (or processes) whose coordinated local action solves global problems. This class of multi-agent system has received an increasing amount of attention in recent years. Their applicability has been proven in many research domains as well as commercial applications. Air traffic control, smart highways, routing, and economic prediction are a handful of applications that make use of this multi-agent paradigm. Decentralized computer security is a natural extension to this list.

There are a number of attractive properties exhibited by multi-agent systems. A multi-agent system can be an effective means of capitalizing on the power of a distributed processing environment. A well-designed multi-agent system will maintain an acceptable level of performance even when portions of the network go down or are compromised. A further advantage is that the processing power of a multi-agent system is extremely scaleable, with agents seamlessly added and removed to the distributed environment as necessary. Because of these attributes, the distributed model lends itself well to real-time reactive control applications, such as insuring the reliability of heterogeneous network elements.

2.2.2 Case Based Reasoning

Case-Based Reasoning (CBR) is the field of AI that deals with the method of solving a current problem by retrieving the solution to a previous similar problem and altering that solution to meet the current needs. Case-based reasoning is a knowledge representation and control methodology based upon previous experiences and patterns of previous experiences. These previous experiences, or "cases" of domain-specific knowledge and action, are used in comparison with new situations or problems. These past methods of solution provide expertise for use in new situations or problems.

Stottler-Henke Associates, Inc. (SHAI) is a pioneer in the development and application of case-based reasoning. We were among the first to employ separate, explicit definitions of similarity which could be used to calculate a quantitative similarity score between a target case and stored cases in the case base. The similarity calculations involve the entire set of features simultaneously, typically with varying degrees of importance. The features may contain simply qualitative (symbol) or quantitative values, or more complicated structures such as objects or lists. SHAI developed rapid retrieval algorithms, [Stottler, Henke, and King, 1989], which could quickly retrieve the most similar case from very large case bases (thousands or millions of cases). The algorithm does not simply build one indexing tree and perform a hierarchical search as most rapid retrieval algorithms do. In effect, our algorithm builds indices on every possible combination of features and performs the retrieval on all indices in parallel.

In MASRR, a case is a set of actions related to collecting data from the environment, communicating with other MASRR agents, and responding to network degrading events. Each case is indexed by features of the situation (i.e., local resource usage statistics, network sniffer statistics, statistics provided by other MASRR agents, and information from the intrusion detection system.) CBR is a very good approach to selecting actions for MASRR, since it is highly configurable and extensible. New cases and corrective actions may be added to the case-base as more is learned about specific problems. Ideally, each MASRR agent will be able to tune itself; although we do not eliminate the possibility of the user explicitly defining or editing cases.

MASRR departs from the traditional CBR model by applying *machine learning* techniques in a couple of ways. First, MASRR does not retrieve individual cases; instead, MASRR retrieves an "action template" which is formed from *clusters* of actual cases. Second, MASRR cannot use traditional CBR indexing methods, since they require that the system have complete information before a case can be retrieved. Instead, MASRR uses a simple *classifier* to perform case lookup. These machine learning techniques are described below.

2.2.3 Machine Learning Techniques: Clustering and Classification

MASRR uses two techniques from the field of machine learning in order to adapt CBR to fit the domain of maintaining network resource reliability. The first of these techniques is *clustering*. Clustering techniques are used to place records into meaningful groups automatically. The intent is for the algorithm to determine useful but hidden classes of network behavior cases that can be generalized into action templates. A well publicized success of a clustering system was the discovery of new stellar spectra classes by the AutoClass program developed at NASA [Cheeseman et al., 1988].

The second machine learning technique the MASRR uses is *classification*. Classification is a basic task in supporting more efficient information retrieval by automating the creation of hierarchical indices. The induction of classifiers from data sets of preclassified instances is a central problem in machine learning. Numerous approaches to this problem are based on various functional representations such as decision trees, decision lists, neural networks, decision graphs, and rules. In MASRR, we have specified Naïve Bayes classifiers as the method for performing retrieval on the Active Template Library. One of the greatest benefits of utilizing Naïve Bayes classifiers is that they offer very accurate results (relative to other approaches) <u>even when information is incomplete</u>. Further, they provide a very useful measure of uncertainty in the classification.

2.3 Literature Search

The proposed work is novel and innovative in a number of ways. In particular, our focus on a completely decentralized, multi-agent, cooperative system for monitoring for and maintaining resource reliability has not been duplicated by any government or private efforts. Below we outline the work that is most similar to ours.

[Thottan & Ji] and **[Hood & Ji]** describe a multi-agent system for performing proactive network management. Each agent uses SNMP ("Simple Network Management Protocol") to collect data from each network element's MIB ("Management Information Base"). This data is *centrally* combined in a Bayesian network in order to learn "normal" behavior for each variable in each MIB. Time series techniques are used to distinguish between a burst of network activity and an actual change in the network signature. In additional to not being decentralized, Ji, Thottan, and Hood's approach does not address the what actions should be taken in response to an observed anomaly.

[Oates] describes a system for performing data mining to identify potential faults in a network element. This system accepts vectors of categorical values describing the state of the network element's behavior as it changes over time and outputs a set of probabilistic rules that describe how features of the network element's state are predictive of future states. Oates presents a useful technique for event correlation that we will extend in Phase II. He does not however make any commitments on how data should be collected or how the predictive information should be used.

[D'haeseleer, Forrest, & Helman] describes a concept of performing distributed anomaly detection based on the mechanisms that exist in immunology. The idea is to create "detectors" that are designed to detect "non-self" entities. Currently, these entities correspond to files (i.e., the system detects changes in the files) and executables (i.e., the system detects sequences of system calls that deviate from the normal pattern). Their approach to network anomaly detection would appear to be decentralized in a way similar to ours, but significant progress has not been demonstrated to date.

[Schroeder & Wagner] and **[Fröhlich et al, 1996]** describes a system of message passing between diagnostic agents on spatially distributed systems. Each agent maintains a detailed model of a small portion of the network, and is assumed to be able to detect all problems that occur on the elements that are covered by the model. The message passing allows an agent to ask its collaborators whether locally detected problems are actually originating from one of the non-local networks. For example, if a message from one sub-network to another gets lost in transit, diagnosis can occur when one of the agents determines that something in its portion of the network is at fault. This represents a very limited diagnostic model and again makes no commitment to what actions should be taken after a problem is diagnosed.

Other related work includes network-based intrusion detection (e.g., **[GrIDS]**) and the detection of malicious routers (**[Pace et al.]**). These projects at UC Davis represent interesting competing techniques to ours. We view these very focused efforts as being subsumed by our approach, but their results will be very valuable in our continuing work.

2.4 Lessons Learned During Phase I

Our Phase I investigation and prototype development has proved very valuable. In our exploration of potential techniques for dealing with network degrading events, we have uncovered features of the domain of network reliability that caused us to reevaluate the focus of this project. In addition, the development of our limited prototype has provided us with a vehicle for testing our preconceptions and highlighting the key challenges in providing this functionality. We list the primary results of our investigation below. Note that each of these findings has been folded into our Phase II approach that is discussed in Section 5.

- **Distributed Case-Based Reasoning (CBR) is inappropriate due to the level of uncertainty in the domain.**

  One of the initial directions of this project was to investigate the possibility of using Distributed CBR to determine how to respond to a network degrading event. In such a CBR system, each MASRR agent would have a set of local "cases," describing past network degrading events and appropriate responses. The MASRR agents would use a description of the current situation to "lookup" a relevant case, and use that to form its response.

  Unfortunately, one of the weaknesses of CBR is that lookup requires *complete information* of a situation. In the domain of maintaining network reliability, a MASRR agent must be able to act before complete information is collected. In fact, in many cases, "complete" information might not even be available for collection.

- **Typical DoS attacks are often single step attacks that are best handled by network element filters.**

  The nature of most network-based DoS attacks involves executing a single directive (often the transmission of a single network packet) that takes advantage of some weakness in a network communication protocol or an operating system. The most effective defense against such an attack is to establish appropriate filters on firewalls, gateways, and routers. Because these DoS attacks execute so rapidly, any attempt to establish these filters during the course of an attack will fail, simply because the offending command will already be on its way to its designated target by the time it is detected.

  For example, consider the "smurf" attack, illustrated in Figure 6. The originator of the smurf attack sends a single broadcast "echo request" network packet to a network, spoofing the source address to that of the "victim." All of the computers within that network send an "echo reply" packet to the victim, often overwhelming that computer's ability to process packets and causing that system to lose network connectivity or to crash. A simple filter on the firewall, router, or other gateway between the originator and the targeted network prevents this attack from occurring.

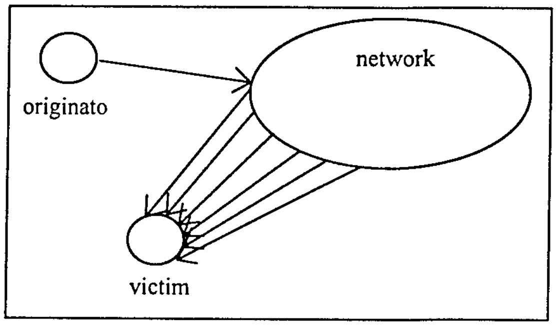


Figure 6. "smurf" DoS attack.

- **However, even when attacks are executed successfully, resource reliability can be maintained by taking measures to alleviate the symptoms.**
  "Homeostasis" can be maintained even in the face of unstopped attacks. That is, network equilibrium over the whole network can be preserved by limiting the side effects that result when particular network elements and/or computer hosts are compromised. For example, affected machines can be temporarily denied network access, alternate machines (usually firewalls) can be "hot swapped" into the network, router filters can be made more restrictive, etc.
- **New attacks, especially those that are variations of known attacks, can be detected and potentially diagnosed through decentralized diagnosis. Diagnosis can also provide appropriate responses to guard against the same attack in the future.**
  Often, new attacks (DoS and otherwise) are adaptations of known attacks, and thus the familiar aspects of the new attacks can still be guarded against, often mitigating their effects. In addition, decentralized network monitoring and information sharing can result in an adequate network-wide description of a new attack, allowing for the development of future responses and/or filters to prevent future occurrences of the attack.

2.5 Technical Feasibility

Several factors contribute to the assurance of the technical feasibility of the proposed MASRR system:

- **Decentralized Diagnosis**. By designing MASRR agents to work using a local view of the network, with locally specified action templates, MASRR will be fully scalable and able to respond to events even with very little information from other parts of the network.
- **Cooperation between Agents**. By designing a robust communication scheme between MASRR agents, protections can be set up across the network in response to events that are detected by individual agents. In addition, individual local views of events are reconciled as they are collected, allowing for increasingly accurate responses and forensics.
- **Robustness**. The design of MASRR focuses on the robustness of individual agents to work under uncertainty, and on the robustness of the multi-agent system to compensate for unexpected events, such as the unavailability of network resources or the crash of a particular host.

2.6 Conclusions

MASRR is different from other computer security tools in the following ways:

- **Decentralized network monitoring, diagnosis, and response**: Unlike many systems, MASRR does not include a central analysis unit, enabling MASRR agents to perform local responses, even when much of the network is compromised. Two key advantages of the decentralized architecture are that it is scalable (i.e., very large networks can be monitored without performance degradation) and extendable(i.e., hosts, network elements, and even whole networks can be added to MASRR's purview with virtually no overhead.)
- **Takes action in response to both known attacks and unexplained anomalies**: Many computer security systems have hard-coded responses to known attacks, but are unable to cope with any events not encoded in their rule systems. Through its decentralized, collaborative agents and the use of action templates, MASRR is also able to respond to unexplained anomalies, such as subverted routing tables and worm-type attacks.
- **Reconciles differing views through data fusion**: An individual MASRR agent can respond with increasing accuracy as information is collected and aggregated from other MASRR agents. Even when network resources are badly compromised, local responses can be applied, allowing rapid recovery from widespread attacks or other network failures.
- **Benefits to Intrusion Detection**: MASRR's design extends intrusion detection monitoring to include network elements/devices (routers, hubs, bridges, gateways, etc). Additionally, intrusion detection can be integrated with routine network management tasks, making optimal use of system administrator efforts while ensuring that intrusion detection systems run using the most up-to-date network information.

3 Phase II Design & Future Work

The goal of SHAI's Multi-Agent System for Resource Reliability is to take network diagnosis and anomaly response away from the current, brittle system of central analysis and move it out into the network, allowing computer hosts and other network elements to maintain local views of network and system activity, thus enabling a system of fast, local decision making. The MASRR system will lead to more robust networks that are better protected against network misconfigurations and the ongoing threat posed by hackers who are constantly discovering new ways of attacking network resources.

3.1 Description of System

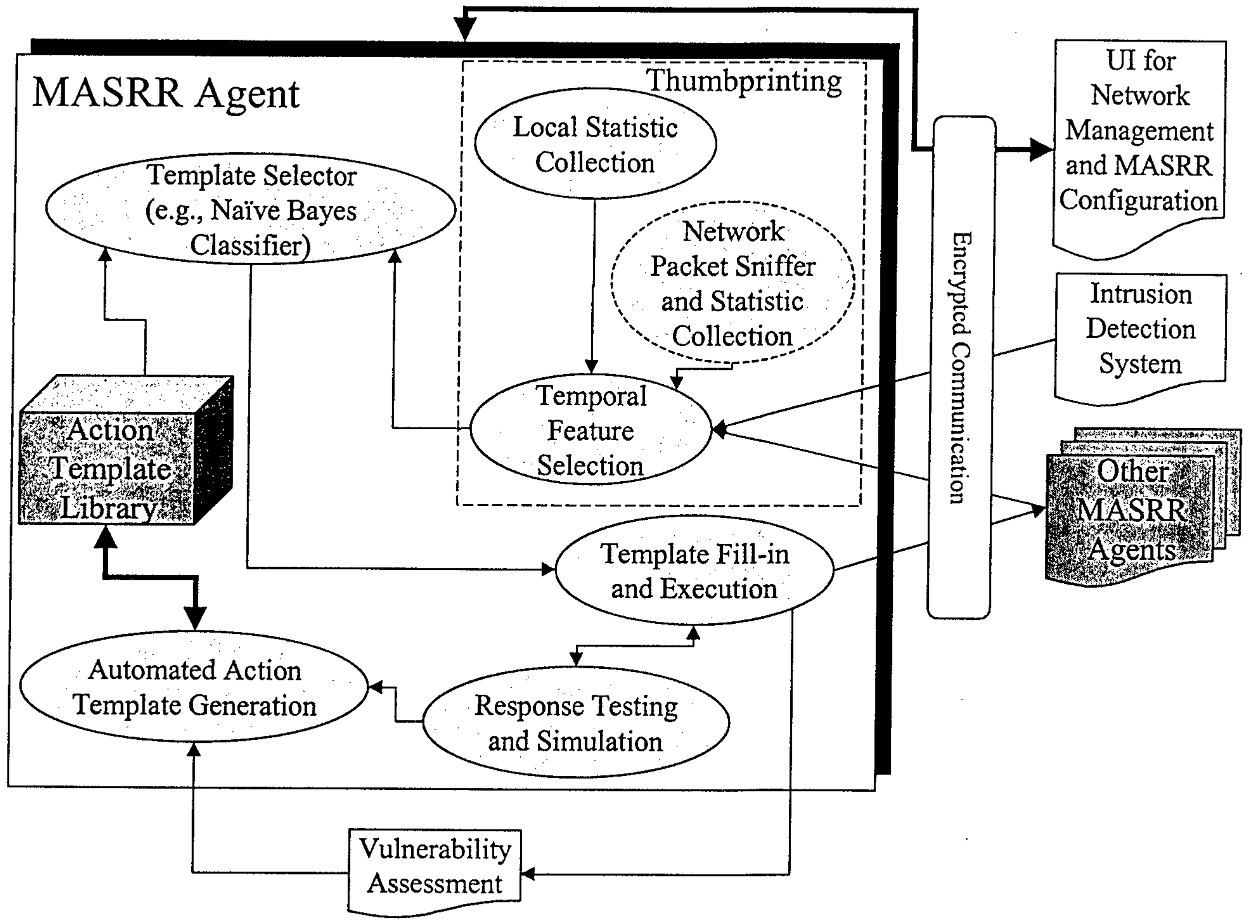

MASRR Agent
Thumbprinting
Local Statistic Collection
Template Selector (e.g., Naïve Bayes Classifier)
Network Packet Sniffer and Statistic Collection
Temporal Feature Selection
Action Template Library
Template Fill-in and Execution
Automated Action Template Generation
Response Testing and Simulation
Encrypted Communication
UI for Network Management and MASRR Configuration
Intrusion Detection System
Other MASRR Agents
Vulnerability Assessment

Figure 7. Phase II System Overview for MASRR.

Figure 7 gives an overview of the information flow within our proposed Phase II MASRR system. As in Phase I, MASRR will consist of individual agents that will be distributed throughout the network, with no central analysis unit. A key issue in the design of these agents will be the development of as-needed, efficient, **secure** communication between agents. MASRR will include an agent communication language that focuses on limiting the impact of inter-agent communication on network bandwidth while supporting the use of secure encryption algorithms [Schneier, 1996] to prevent the possibility of forged messages being introduced to the system.

Each MASRR agent will maintain a set of **thumbprints**, representing "normal" system and network activity. These thumbprints will encode such features as system load, memory and hard disk usage, and type and quantity of network traffic. When local behavior deviates significantly from the local thumbprint data, the **Template Selector** will use machine learning techniques (e.g., Naïve Bayes Classifiers) to use partial knowledge to select an **action template** from a local **action template library**. **Temporal feature selection and construction techniques** will be applied to the collected statistics to improve the quality of local thumbprint data, and to make the collected data more usable by the template selection mechanism.

Once an action template is selected, a MASRR agent will use the collected data to **fill in** the template and **execute** it. Execution may include: further evidence collection, sending the data encoded in the action template to other MASRR agents via **secure encrypted communication**, sending an encrypted request for current statistics from SHAI's **intrusion detection system**, ICE, sending an alert to the system administrator, and **responding to network degrading events**.

One possibility for MASRR's response mechanism is to provide a real-time **Response Simulation** system for testing multiple candidate responses and choosing the one with the greatest payoff while minimizing the impact on users. This simulation and testing system will only be possible in situations where MASRR either does not need to respond to an event immediately, or when a temporary response can be used to slow down events. Simulation will take place using SHAI's proprietary "Projective Simulation" technique. The ability to test candidate responses will also enable MASRR to **automatically generate new action templates**. In addition, SHAI's **Strategic Vulnerability Assessment** mechanism will provide further support for generating new action templates by identifying particular vulnerabilities in the network that enabled an attack to occur.

The final component of the Phase II MASRR system will be a **user interface**, which will give system administrators the ability to centrally configure the distributed MASRR agents. In addition, this user interface may provide functionality for network management, including the ability to configure routers and other network elements, and institute filtering and auditing policies. Any directives sent from this user interface out into the network will use the same encrypted communication as the inter-agent communication, so as to prevent hackers from overriding network policies.

3.2 Phase II Technical Objectives

Phase II research and development will build on the significant progress made in Phase I and result in a complete prototype of MASRR, a system for monitoring heterogeneous networks, diagnosing network degrading events, and establishing repair strategies when possible. The primary goals of the Phase II research are to:

1. **Elaborate the key functionality of the MASRR system**:
   a. We will refine MASRR's thumbprint representations and statistics collection.
   b. We will develop a comprehensive design for MASRR's action templates, including representations for thumbprint deviations, agent-to-agent communication, and action template selection.
   c. We will investigate the potential for semi-automatic construction of local action template libraries by combining machine learning techniques (e.g., classification, clustering) with dependency analysis. These techniques may use data from a variety of sources, including audit logs and commercial network modeling tools (e.g., [Retriever], [SecureScanner]).
   d. We will explore the potential benefit of applying SHAI's Projective Simulation techniques to the task of testing candidate responses to network degrading events.
   e. We will apply temporal feature selection techniques to the statistical data collected by individual MASRR agents.

f. We will develop a user interface for configuring individual MASRR agents, and potentially for performing related network management tasks.

2. **MASRR will be integrated with our proprietary intrusion detection system, ICE, and our vulnerability detection prototype, SecurE.** ICE will provide statistics on the likelihood that a known attack is occurring on the network, allowing local MASRR agents to select responses most able to thwart such an attack; in return, MASRR agents will provide statistical data to help ICE confirm or refute ongoing attack hypotheses. SecurE will provide information as to the source of the vulnerability that allowed a current attack, providing data for the development of new action templates.

3. **MASRR's effectiveness and scalability will be evaluated** by using test data that has been generated for evaluating the effectiveness of intrusion detection systems. An example of such data is available at http://www.ll.mit.edu/SST/ideval/index.html.

4 Commercialization Plans

The number of computer networks utilized in the private and public sectors is growing exponentially. Along with this increasing dependence on information technology, is a growing worry about security and the reliability of network resources. This concern offers a unique opportunity for the marketing of tools that can increase the reliability of network resources. We feel that the reliability provided by a complete MASRR system will fit the needs of a great number of institutions.

Central to our research effort is to provide the reliability of network resources across complex heterogeneous networks. Throughout our Phase I and Phase II work, we will work towards dual purposes. We intend to both develop a tool to suit DARPA's specific requirements, as well as, work towards the creation of a more general resource reliability system. The potential market for such a tool is huge, with potential clients including any corporation operating a computer network.

There are two types of products for commercialization. First, we can market our general MASRR tool and allow the customer to tailor it to their particular environment. From our experience with marketing ESTEEM, SHAI's commercially available case based reasoning (CBR) application development tool, we understand how to successfully market customizable products. Articles and advertisements in relevant journals and magazines, exhibition booths and tutorials at relevant conferences, direct mailings using lists purchased from relevant organizations, involvement in industry associations, and a strong World Wide Web page are all important aspects. These methods generated an order of magnitude more funds than SHAI received in SBIR funds.

Second, we can customize MASRR for individual clients and markets - tailoring specialized resource reliability agents for specific network architectures and reliability concerns. Because our system will be designed for quick domain application, it could be employed in new domains with little development time. Marketing such software specialization services is similar to SHAI's core business of marketing AI research and development services, at which we are very successful.

5 References

[Bellovin, 1989] "Security Problems in the TCP/IP Protocol Suite"; *Computer Communications Review*, Volume 19, Number 2; April 1989; pp.32-48.

[Bradshaw, 1997] *Software Agents*, AAAI Press / MIT Press, 1997.

[CERT CA-97.28] "CERT Advisory CA-97.28: IP Denial-of-Service Attacks", http://www.cert.org/CA-97.28.Teardrop_Land.html

[CERT CA-98.01] "CERT Advisory CA-9.8.01: 'smurf' IP Denial-of-Service Attacks". http://www.cert.org/CA-98.01.smurf.html

[Cheeseman et al. 1988] "AutoClass: A Bayesian Classification System," 5th International Conference on Machine Learning, pp. 54, 1988.

[D'haeseleer, Forrest, & Helman] "A distributed approach to anomaly detection," Submitted to ACM Transactions on Information System Security (1997).

[Fröhlich et al.] Fröhlich, Móra, Nejdl, Michael Schroeder, " Diagnostic Agents for Distributed Systems," In *Proceedings of ModelAge97*.

[GrIDS] Staniford-Chen, Cheung, Crawford, Dilger, Frank, Hoagland, Levitt, Wee, Yip, & Zerkle. "GrIDS - A Graph Based Intrusion Detection System for Large Networks," *Proceedings of the 19th National Information Systems Security Conference*, Baltimore, MD, Oct. 1996, 361 – 370

[Hood & Ji] "Intelligent Agents for Proactive Fault Detection," IEEE Internet Computing, Vol. 2, No. 2, March/April 1998.

[Keogh & Pazzani] (1999). Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches. Uncertainty 99, 7th. Int'l Workshop on AI and Statistics, Ft. Lauderdale, Florida, 225-230.

[Oates] "Fault Identification in Computer Networks: A Review and a New Approach", Computer Science Department Technical Report 95-113, University of Massachusetts, Amherst, 1995.

[Pace et al.] Pace, Wee, Mukherjee, & Olsson "Detection of Subverted Routers in an Internet: An audit based approach", http://olympus.cs.ucdavis.edu/nra/reports/Pace-Wee-Muhkerjee-Olsson.pdf.

[Retriever] Retriever by L-3 Communications http://www.l-3com.com/ http://www.l3security.com/

[Schneier, 1996] *Applied Cryptography*, John Wiley & Sons, Inc. 1996.

[Schroeder & Wagner] "Distributed Diagnosis by Vivid Agents," 1st International Conference on Autonomous Agents, ACM Press, 1997.

[SecureScanner] SecureScanner (formerly NetSonar) by Cisco Systems http://www.cisco.com/warp/public/cc/cisco/mkt/security/nsonar/index.shtml

[Stottler, Henke, and King, 1989] "Rapid Retrieval Algorithms for Case-Based Reasoning," *IJCAI-89 Eleventh International Joint Conference on Artificial Intelligence - Proceedings, Volume 1:233-237,* Detroit, MI.

[Thottan & Ji] "Proactive Anomaly Detection Using Distributed Intelligent Agents", IEEE Network, Special Issue on Network Management. 1998.

[Tomlin, Pappas, and Sastry, 1997] "Conflict Resolution for Air Traffic Management: a Study in Multi-Agent Hybrid Systems", *1997 IEEE Conference on Decision and Control*, 1997.

Appendix A – Demonstration Sequences

For the final briefing of this contract, we prepared two demonstrations. The first demonstration showed how a MASRR agent would process data from a network sniffer in the formation of thumbprints and the subsequent monitoring for anomalous network behavior. For this demonstration, we used the "Sample Data" tcpdump output file obtained from http://www.ll.mit.edu/IST/ideval/index.html. The results of this demonstration are shown in Figure 3.

The second demonstration included six sequences. Three of the sequences showed how certain attacks would transpire over a network without any protection. The other three showed how multiple MASRR agents would work together to limit the effects of those same attacks. To perform this demonstration, we constructed an interface using a PowerPoint presentation representing the network (see Figure 5). Each node in the network diagram changed color when some aspect of the attack manifested (including when a host or network element crashed), when a MASRR agent detected some suspicious activity and sent a communication to other MASRR agents, and when a MASRR agent received and replied to a message.

In the demonstration sequences where MASRR agents were monitoring the network, actual executions of the MASRR prototype were created (running as a simulation on a single computer). To correspond to the network shown in Figure 5, we executed four instances of MASRR corresponding to the routers, one instance of MASRR corresponding to the firewall, and twenty-four instances of MASRR corresponding to the hosts. Communication between the MASRR agents was performed via simple socket connections.

The six demonstration sequences were:

Smurf DoS attack (no MASRR monitoring) – In this attack, a Smurf attack is launched from Host X.10.4 by sending a broadcast Echo Request with the Source forged to X.44.23 (the "victim"), and the Destination set to X.30.255 (broadcast address). This sequence proceeded as follows:

1. X.10.4 sends the Echo Request packet with the forged source address to X.30.255.
2. The packet travels through Router 1.
3. All machines in the sub-network X.30.XXX receive the Echo Request packet, and send an Echo Reply packet to X.44.23. (Note: In the PowerPoint picture, there are only 6 machines in this sub-network; however, there could potentially be hundreds of machines.)
   The network sniffer on X.30.37 records an increase in the Echo Reply traffic in the sub-network.
4. The Echo Reply packets travel through Router 1.
5. The Echo Reply packets travel through Router 2.
6. The Echo Reply packets travel through Router 3.
7. The Echo Reply packets arrive at Host X.44.23, which subsequently crashes. Meanwhile, the network sniffer on X.44.38 records an increase in the Echo Reply traffic in the sub-network.

8. Resubmit traffic on Router 3 increases, causing a deterioration in network bandwidth.

End: Eventually, all the packets time out, and network behavior returns to normal. X.44.23 must be rebooted.

Smurf DoS attack (router filter in place) – Again, a Smurf attack is launched from Host X.10.4 by sending a broadcast Echo Request with the Source forged to X.44.23 (the "victim"), and the Destination set to X.30.255 (broadcast address). This sequence proceeded as follows:

1. X.10.4 sends the Echo Request packet with the forged source address to X.30.255.
2. The packet travels through Router 1. Router 1 filter catches the spoofed source address (i.e., the source address does not exist on the machines on that port), and drops the packet.

Note: If no such filter were in place, MASRR could mitigate the effects of the Smurf attack at a variety of places along the attack path. The "Unknown Attack" described at the end of this appendix is actually a Smurf attack, but working under conditions where the attack has not been diagnosed and preventive filters have not been put in place.

Internet Worm (no MASRR monitoring) – In this attack, the Internet Worm has attacked Host X.10.4. This sequence proceeded as follows:

1. X.10.4 is infected by the Internet Worm. As the Worm multiplies, load average and memory usage goes up.
2. The Worm has infected Host X.10.1. As the Worm multiplies load average and memory usage goes up.
   In addition, the Intrusion Detection system on Host X.10.17 detects an attempt to exploit the sendmail bug.
3. The Intrusion Detection system on Host X.10.17 detects an attempt to exploit the finger bug.
4. The Worm has infected Host X.10.17 (apparently the finger bug wasn't patched, even though the intrusion detection system knew to monitor it). As the Worm multiplies load average and memory usage goes up.
   Meanwhile, Host X.10.4 crashes.
5. Host X.10.1 crashes.
   The Worm has infected Host X.30.23. As the Worm multiplies load average and memory usage goes up.
   In addition, the Intrusion Detection system on Host X.30.40 detects an attempt to exploit the sendmail bug.
6. The Worm has infected Host X.30.33. As the Worm multiplies load average and memory usage goes up.
   In addition, the Intrusion Detection system on Host X.30.40 detects an attempt to exploit the finger bug.
7. Host X.10.17 crashes.
   Host X.30.23 crashes.

8. Host X.30.33 crashes.

End: This is the end of the demonstration. However, in the actual occurrence of the Internet Worm, it undoubtedly would continued indefinitely, with far more widespread results.

Internet Worm (MASRR monitoring in place) – Again, the Internet Worm has attacked Host X.10.4. (Note: This scenario is also described in Section 2.1.1.) This sequence proceeded as follows:

1. X.10.4 is infected by the Internet Worm. As the Worm multiplies, load average and memory usage goes up.
   MASRR agent broadcasts thumbprint info to other hosts in the sub-network.
2. The Worm has infected Host X.10.1. As the Worm multiplies load average and memory usage goes up.
   In addition, the Intrusion Detection system on Host X.10.17 detects an attempt to exploit the sendmail bug.
   Hosts X.10.1,3,10,13, and 17 receive message from X.10.4.
   Hosts X.10.3,10,13 reply with no information.
   Host X.10.1 replies with "load average and memory usage is high"
   Host X.10.17 replies with Intrusion Detection message.
3. The Intrusion Detection system on Host X.10.17 detects an attempt to exploit the finger bug.
   Host X.10.4 sends aggregated information to Router 1.
4. The Worm has infected Host X.10.17 (apparently the finger bug wasn't patched, even though the intrusion detection system knew to monitor it). As the Worm multiplies load average and memory usage goes up.
   Meanwhile, Host X.10.4 crashes
   Router 1 receives message from Host X.10.4
   Router 1 restricts network traffic on sub-network X.10.1-20
   Router 1 sends a message to the system administrator.
5. Host X.10.1 crashes.
   Host X.10.17 crashes.

End: This is the end of the demonstration. Other machines in the network X.10.1-20 may go down, but the rest of the network is protected and continues to function normally.

Unknown attack (no MASRR monitoring) – This is actually the Smurf attack described as the first attack in this set, but working under conditions where the attack has not been diagnosed and preventive filters have not been put in place.

Unknown attack (MASRR monitoring in place) – An unknown attack is launched from Host X.10.4 by sending a broadcast Echo Request with the Source forged to X.44.23 (the "victim"), and the Destination set to X.30.255 (broadcast address). This sequence proceeded as follows:

1. X.10.4 sends the Echo Request packet with the forged source address to X.30.255.

2. The packet travels through Router 1.
   The MASRR agent detects the anomaly (i.e., spoofed source address), and sends a message to MASRR agents in the X.30.XXX sub-network, and to the MASRR agent on X.44.23.

3. All machines in the sub-network X.30.XXX receive the Echo Request packet, and send an Echo Reply packet to X.44.23. (Note: In the PowerPoint picture, there are only 6 machines in this sub-network; however, there could potentially be hundreds of machines.)
   The network sniffer on X.30.37 records an increase in the Echo Reply traffic in the sub-network.
   The MASRR agent on X.30.37 responds to Router 1 with network statistics. All other MASRR agents on the sub-network reply with no information.
   The MASRR agent on X.44.23 replies with no information.

4. The Echo Reply packets travel through Router 1. **The MASRR agent on Router 1 detects an increase in Echo Reply traffic.**
   The MASRR agent on Router 1 aggregates its local information with the reply from X.30.37, and sends a message to Router 2 and Router 3.

5. The Echo Reply packets travel through Router 2. **The MASRR agent on Router 2 detects an increase in Echo Reply traffic.**
   The MASRR agent on Router 2 responds with its local information.

6. The Echo Reply packets travel through Router 3. **The MASRR agent on Router 3 detects an increase in Echo Reply traffic.**
   The MASRR agent on Router 3 responds with its local information.

7. The Echo Reply packets arrive at Host X.44.23, which subsequently crashes. Meanwhile, the network sniffer on X.44.38 records an increase in the Echo Reply traffic in the sub-network.
   The MASRR agent on X.44.38 sends a message to the other MASSR agents on the sub-network.

8. Resubmit traffic on Router 3 increases, causing a deterioration in network bandwidth.
   The MASRR agents on the X.44.XXX sub-network respond with no information. The MASRR agent on X.44.23 does not respond.
   The MASRR agent on X.44.38 aggregates the LAN statistics, and sends a message to the MASRR agent on Router 3.

9. **The MASRR agent on Router 3 prepares new response for the MASRR agent on Router 1, indicating that X.44.23 has crashed, and including new statistics on the resubmit traffic.**

10. **The MASRR agent on Router 1 responds by dropping all resubmit traffic destined for X.44.23. Router 3 recovers, and network bandwidth is restored.**